

PERMUTATIONS OF CONTEXT-FREE AND INDEXED LANGUAGES

TARA BROUGH, LAURA CIOBANU, AND MURRAY ELDER

ABSTRACT. We consider the cyclic closure of a language, and its generalisation to the operators C^k introduced by Brandstädt. We prove that the cyclic closure of an indexed language is indexed, and that if L is a context-free language then $C^k(L)$ is indexed.

1. INTRODUCTION

The *cyclic closure* of a language L is the language

$$\text{cyc}(L) = \{w_2w_1 \mid w_1w_2 \in L\}.$$

It is easy to show that the classes of regular, context-sensitive and recursively enumerable languages are closed under this operation. Maslov and independently Oshiba [6, 7] proved that the class of context-free languages is also closed under this operation. In this article we show that in addition, the class of indexed languages is closed under taking cyclic closure.

Brandstädt [2] generalised the notion of cyclic closure to a family of operators on languages C^k for $k \in \mathbb{N}$ by defining

$$C^k(L) = \{w_{\sigma(1)} \dots w_{\sigma(k)} \mid w_1 \dots w_k \in L, \sigma \in S_k\},$$

where S_k is the set of permutations on k letters. So $C^2(L)$ is exactly the cyclic closure. He proved that if L is context-free and $k \geq 3$ the language $C^k(L)$ is not context-free in general, while it is always context-sensitive. Here we sharpen this result by showing that $C^k(L)$ is indexed.

A natural generalisation of context-free and indexed languages was given by Damm and co-authors in [3, 4], where they defined the OI- and IO-hierarchies of languages built out of automata or grammars that extended the pushdown automata and indexed grammars, respectively. They define level- n grammars inductively, allowing the flags at level n to carry up to n levels of parameters in the form of flags. Thus level-0 grammars generate context-free languages, and level-1 grammars produce indexed languages.

Date: January 6, 2015.

2010 Mathematics Subject Classification. 20F65; 68Q45.

Key words and phrases. indexed grammar; context-free grammar; cyclic closure.

Research supported by Australian Research Council grant FT110100178, Swiss National Science Foundation Professorship FN PP00P2-144681/1 and London Mathematical Society Scheme 4 grant 41348.

We conjecture that the class of level- n languages is closed under cyclic closure, and also that if L is an level- n language then $C^k(L)$ is an level- $(n+1)$ language. This paper is the first step in proving this conjecture and completing the picture of cyclic closure and permutation operators for the OI- and IO-hierarchies.

2. PRELIMINARIES

2.1. Permutation operators. Brandstädt [2] defined the language $C^k(L)$ to be the set of all words obtained from L by permutating k subwords according to some permutation. In this article we specialise the definition to individual permutations as follows.

Definition 1 (Permutation operator). *Let k be a positive integer, $\sigma \in S_k$ a permutation on $\{1, \dots, k\}$, and $L \subseteq \Sigma^*$ a language over a finite alphabet. The language $\sigma(L)$ is defined by*

$$\sigma(L) = \{w_{\sigma(1)} \cdots w_{\sigma(k)} \mid w_1 \cdots w_k \in L\}.$$

If $w = w_{\sigma(1)} \cdots w_{\sigma(k)} \in \sigma(L)$ where $w_1 \cdots w_k \in L$, some subwords w_i could be empty. We can write $w_1 \cdots w_k$ as $u_1 \cdots u_\ell$ for some $\ell \leq k$, where each u_i is equal to some non-empty w_j . Then there is a permutation $\tau \in S_\ell$ (called a *subpattern* of σ) such that $w = u_{\tau(1)} \cdots u_{\tau(\ell)}$.

For $\tau \in S_\ell$, define

$$L_\tau = \{w_{\tau(1)} \cdots w_{\tau(\ell)} \mid w_1 \cdots w_\ell \in L, w_i \neq \varepsilon \ (1 \leq i \leq \ell)\}.$$

Then $\sigma(L) = \bigcup_\tau L_\tau$ with τ ranging over all subpatterns of σ . Thus if \mathcal{C}_1 and \mathcal{C}_2 are two language classes, with \mathcal{C}_2 closed under finite union, and we wish to show that $\sigma(L) \in \mathcal{C}_2$ for all $L \in \mathcal{C}_1$ and $\sigma \in S_k$, it suffices to show that $L_\tau \in \mathcal{C}_2$ for all $L \in \mathcal{C}_1$ for all $\tau \in \bigcup_{1 \leq \ell \leq k} S_\ell$.

For any language L and $k \in \mathbb{N}$, we have

$$C^k(L) = \bigcup_{\sigma \in S_k} \sigma(L) = \bigcup_{1 \leq \ell \leq k} \bigcup_{\sigma \in S_\ell} L_\sigma.$$

Note that the language L_τ does not in general contain L as a sublanguage, whereas $\sigma(L)$ contains L since we may take all but one subword to be empty.

2.2. Indexed languages. We define an indexed language to be one that is generated by the following type of grammar:

Definition 2 (Indexed grammar; Aho [1]). *An indexed grammar is a 5-tuple $(\mathcal{N}, \mathcal{T}, \mathcal{I}, \mathcal{P}, \mathbf{S})$ such that*

- (1) $\mathcal{N}, \mathcal{T}, \mathcal{I}$ are three mutually disjoint sets of symbols, called nonterminals, terminals and indices (or flags) respectively.
- (2) $\mathbf{S} \in \mathcal{N}$ is the start symbol.
- (3) \mathcal{P} is a finite set of productions, each having the form of one of the following:
 - (a) $\mathbf{A} \rightarrow \mathbf{B}^f$.
 - (b) $\mathbf{A}^f \rightarrow v$.

(c) $\mathbf{A} \rightarrow u$.
 where $\mathbf{A}, \mathbf{B} \in \mathcal{N}$, $f \in \mathcal{I}$ and $u, v \in (\mathcal{N} \cup \mathcal{T})^*$.

The language defined by an indexed grammar is the set of all strings of terminals that can be obtained by successively applying production rules starting from a rule which has the start symbol \mathbf{S} on the left. Production rules operate as follows. Let $\mathbf{A} \in \mathcal{N}$, $\gamma \in \mathcal{I}^*$ and suppose \mathbf{A}^γ appears in some string.

- (1) applying $\mathbf{A} \rightarrow \mathbf{B}^f$ replaces \mathbf{A}^γ by $\mathbf{B}^{f\gamma}$
- (2) if $\gamma = f\delta$ with $f \in \mathcal{I}$, applying $\mathbf{A}^f \rightarrow \mathbf{B}a\mathbf{C}$ replaces \mathbf{A}^γ by $\mathbf{B}^\delta a \mathbf{C}^\delta$
- (3) applying $\mathbf{A} \rightarrow \mathbf{B}a\mathbf{C}$ replaces \mathbf{A}^γ by $\mathbf{B}^\gamma a \mathbf{C}^\gamma$.

We call the operation of successively applying productions starting from one which has the start symbol \mathbf{S} on the left and terminating at a string $u \in \mathcal{T}^*$ a *derivation* of u . We use the notation \Rightarrow to denote a sequence of productions within a derivation, and call such a sequence a *subderivation*.

Definition 3 (Normal form). *An indexed grammar $(\mathcal{N}, \mathcal{T}, \mathcal{I}, \mathcal{P}, \mathbf{S})$ is in normal form if*

- (1) *the start symbol only appears on the left side of a production,*
- (2) *productions are of the following type:*
 - (a) $\mathbf{A} \rightarrow \mathbf{B}^f$
 - (b) $\mathbf{A}^f \rightarrow \mathbf{B}$
 - (c) $\mathbf{A} \rightarrow \mathbf{B}\mathbf{C}$
 - (d) $\mathbf{A} \rightarrow a$*where $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{N}$, $f \in \mathcal{I}$ and $a \in \mathcal{T}$.*
- (3) \mathcal{I} *contains a special ‘end-of-flag’ symbol $\$$, and the start symbol begins with the string $\$$ in its flag. The symbol $\$$ is otherwise never used in any $\mathbf{A} \rightarrow \mathbf{B}^f$ production.*

An indexed grammar can be put into normal form as follows. Introduce a new nonterminal \mathbf{S}_0 , a production $\mathbf{S}_0 \rightarrow \mathbf{S}$, and declare \mathbf{S}_0 to be the new start symbol. This ensures condition (1). For each production $\mathbf{A}^f \rightarrow v$ with $v \notin \mathcal{N}$, introduce a new nonterminal \mathbf{B} , add productions $\mathbf{A}^f \rightarrow \mathbf{B}$, $\mathbf{B} \rightarrow v$, and remove $\mathbf{A}^f \rightarrow v$. By the same arguments used for Chomsky normal form, each production $\mathbf{A} \rightarrow u$ without flags can be replaced by a set of productions of type 2c and 2d above. Instead of beginning derivations with the start symbol \mathbf{S}_0 (with empty flag), begin with $\mathbf{S}_0^\$$. (Introducing the symbol $\$$ in this way into an existing indexed grammar is pointless, but harmless. For constructing new grammars, however, it is often very useful to have a way of telling when a flag is ‘empty’.)

In an indexed grammar in normal form, every nonterminal in a derivation has a flag of the form $\gamma\$$ where $\gamma \in \mathcal{I}^*$. The symbol $\$$ is removed only by productions of type 2d.

2.3. Tree-shapes and parse tree skeletons. A *parse tree* in an indexed or context-free grammar is a standard way to represent a derivation in the

grammar, see for example [5]. In this paper, all trees will be rooted, and will be regarded as being drawn in the plane, with the root at the top, and with a fixed orientation. For a tree T , the *shape* of T is the tree \hat{T} obtained from T as follows:

- (1) add an edge to the root vertex, so that the root has degree 1
- (2) delete all vertices of degree 2

A *tree-shape* is hence a tree with no vertices of degree 2, and root degree 1. For example, the two possible tree-shapes with 3 leaves are shown in Figure 1.

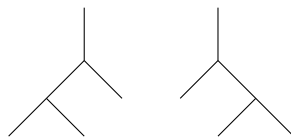


FIGURE 1. Possible tree-shapes with 3 leaves.

Let P be a parse tree in some grammar and F a subtree of P (with all labels preserved). Let B be the 1-neighbourhood¹ of F , and suppose F has tree-shape T . We call B a *T-skeleton* of P , and F the *frame* of B .

If Γ is a grammar, then we call a tree B labelled by symbols from Γ a *T-skeleton in Γ* if B is a T -skeleton of some parse tree in Γ . For example, if T is the first of the tree-shapes in Figure 1, then Figure 2 is a T -skeleton in some indexed grammar, with the frame F marked in bold.

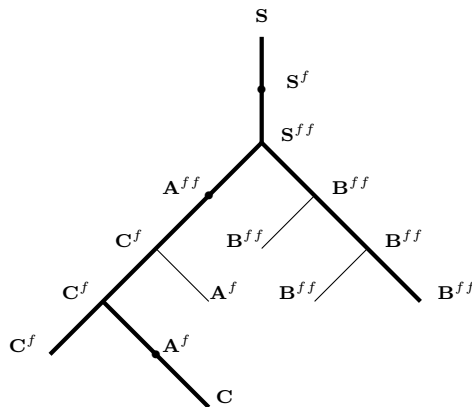


FIGURE 2. A T -skeleton in an indexed grammar.

If F is a path, then the shape of F is an edge, and so we call a skeleton with frame F an *edge-skeleton*.

¹the subtree of P consisting of all edges with at least one end vertex in F

3. MAIN RESULTS

Brandstädt proved that the classes of regular, context-sensitive and recursively enumerable languages are closed under the operation C^k for all k [2]. We start by reproving Theorem 1 of [2], modified for $\sigma(L)$.

Lemma 4. *If L is regular then $\sigma(L)$ is regular for any fixed permutation σ .*

Proof. Assume L is the language of a finite state automaton M with start state q_{start} and single accept state q_{accept} , and $\sigma \in S_k$. For each $(k-1)$ -tuple of states $\mathbf{q} = (q_{j_1}, \dots, q_{j_{k-1}})$, define k automata $M_1^{\mathbf{q}}, \dots, M_k^{\mathbf{q}}$ as follows. Let $M_1^{\mathbf{q}}$ be a copy of M with start state q_{start} and accept state q_{j_1} . For $1 \leq s < k-1$ let $M_{s+1}^{\mathbf{q}}$ be a copy of M with start state q_{j_s} and accept state $q_{j_{s+1}}$. Let $M_k^{\mathbf{q}}$ be a copy of M with start state $q_{j_{k-1}}$ and accept state q_{accept} .

Define $\overline{M_i^{\mathbf{q}}}$ to be the language accepted by the automaton $M_i^{\mathbf{q}}$, and let $L_{\mathbf{q}}$ be the concatenation

$$\overline{M_1^{\mathbf{q}}} \overline{M_2^{\mathbf{q}}} \dots \overline{M_k^{\mathbf{q}}}.$$

Then $L_{\mathbf{q}}$ accepts precisely the words in L that label a path in M from q_{start} to q_{accept} that passes the intermediate states from \mathbf{q} . It follows that $L = \bigcup_{\mathbf{q}} L_{\mathbf{q}}$. Now define $L_{\mathbf{q}}^{\sigma}$ to be the concatenation

$$\overline{M_{\sigma(1)}^{\mathbf{q}}} \overline{M_{\sigma(2)}^{\mathbf{q}}} \dots \overline{M_{\sigma(k)}^{\mathbf{q}}}.$$

Then $w \in L_{\mathbf{q}}^{\sigma}$ if and only if $w = w_{i_1} \dots w_{i_k}$ and

$$w_{\sigma^{-1}(i_1)} \dots w_{\sigma^{-1}(i_k)} \in L_{\mathbf{q}}.$$

It follows that $\sigma(L) = \bigcup_{\mathbf{q}} L_{\mathbf{q}}^{\sigma}$. \square

Maslov and independently Oshiba [6, 7] proved that the cyclic closure of a context-free language is context-free. A sketch of a proof of this fact is given in the solution to Exercise 6.4 (c) in [5], and we generalise the approach taken there to show that the class of indexed languages is also closed under the cyclic closure operation.

Theorem 5. *If L is indexed, then $\text{cyc}(L)$ is indexed.*

Proof. The idea of the proof is to take the parse-tree of a derivation of $w_1 w_2 \in L$ in Γ and “turn it upside down”, using the leaf corresponding to the first letter of the word w_2 as the new start vertex.

Let $\Gamma = (\mathcal{N}, \mathcal{X}, \mathcal{I}, \mathcal{P}, \mathbf{S})$ be an indexed grammar for L in normal form. If $w = a_1 \dots a_n \in L$ with $a_i \in \mathcal{X}$ and we wish to generate the cyclic permutation $a_k \dots a_n a_1 \dots a_{k-1}$ of w , take some parse tree for w in Γ and draw the unique path F from the start symbol $\mathbf{S}^{\$}$ to a_k . Consider the edge-skeleton of this parse tree with frame F .

In the example given in Figure 3, the desired word $a_k \dots a_n a_1 \dots a_{k-1}$ can be derived from the string $a_k \mathbf{A}_3^{f\$} \mathbf{A}_4^{f\$} \mathbf{A}_1^{\$} \mathbf{A}_2^{gf\$}$, using productions in \mathcal{P} .

Therefore we wish to enlarge the grammar to generate all strings

$$a_k \mathbf{A}_{k+1}^{w_{k+1}} \dots \mathbf{A}_n^{w_n} \mathbf{A}_1^{w_1} \dots \mathbf{A}_{k-1}^{w_{k-1}},$$

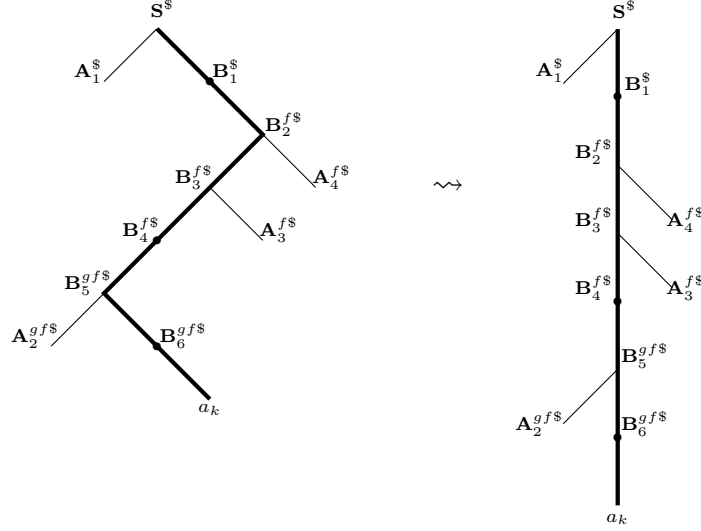


FIGURE 3. Edge-skeleton in an indexed grammar. The right-hand version is the same tree as the left-hand one, but ‘straightened’ along the path from \mathbf{S}^s to a_k .

where $\mathbf{A}_1^{w_1}, \dots, \mathbf{A}_{k-1}^{w_{k-1}}$ are the labels of the vertices lying immediately to the left of F (in top to bottom order), and $\mathbf{A}_{k+1}^{w_{k+1}}, \dots, \mathbf{A}_n^{w_n}$ are the labels of the vertices lying immediately to the right of F (in bottom to top order). We do this by introducing new ‘hatted’ nonterminals, with which we label all the vertices along the path F , and new productions which are the reverse of the old productions ‘with hats on’. By first nondeterministically guessing the flag on the nonterminal immediately preceding a_k , we are able to essentially generate the edge-skeleton in reverse.

The grammar for $\text{cyc}(L)$ is given by $(\mathcal{N}', \mathcal{T}', \mathcal{I}', \mathcal{P}', \mathbf{S}_0)$, where $\mathcal{T}' = \mathcal{T}$, $\mathcal{I}' = \mathcal{I}$, \mathbf{S}_0 is the new start symbol, and \mathcal{N}' and \mathcal{P}' are as follows. Let $\hat{\mathcal{N}}$ be the set of symbols obtained from \mathcal{N} by placing a hat on them. Then $\mathcal{N}' = \mathcal{N} \cup \hat{\mathcal{N}} \cup \{\mathbf{S}_0, \tilde{\mathbf{S}}\}$ is the new set of nonterminals.

The productions \mathcal{P}' are obtained as follows:

- keep all the old productions from \mathcal{P} .
- add productions $\mathbf{S}_0 \rightarrow \mathbf{S}$, $\mathbf{S}_0 \rightarrow \tilde{\mathbf{S}}$, $\hat{\mathbf{S}}^s \rightarrow \varepsilon$
- for each $f \in \mathcal{I}$, add a production $\tilde{\mathbf{S}} \rightarrow \tilde{\mathbf{S}}^f$
- for each production $\mathbf{A} \rightarrow a$ in \mathcal{P} , add a production $\tilde{\mathbf{S}} \rightarrow a\hat{\mathbf{A}}$
- for each production $\mathbf{A} \rightarrow \mathbf{B}^f$ in \mathcal{P} , add a production $\hat{\mathbf{B}}^f \rightarrow \hat{\mathbf{A}}$
- for each production $\mathbf{A}^f \rightarrow \mathbf{B}$ in \mathcal{P} , add a production $\hat{\mathbf{B}} \rightarrow \hat{\mathbf{A}}^f$
- for each production $\mathbf{A} \rightarrow \mathbf{BC}$ in \mathcal{P} , add productions $\hat{\mathbf{B}} \rightarrow \mathbf{C}\hat{\mathbf{A}}$ and $\hat{\mathbf{C}} \rightarrow \hat{\mathbf{A}}\mathbf{B}$.

Note that the new grammar is no longer in normal form. Also note that the only way to remove the hat symbol is to apply the production $\hat{\mathbf{S}}^s \rightarrow \varepsilon$.

We will show by induction that in this new grammar,

$$(1) \quad \mathbf{A}^w \Rightarrow \mathbf{A}_1^{w_1} \dots \mathbf{A}_i^{w_i} \dots \mathbf{A}_n^{w_n}$$

if and only if

$$(2) \quad \hat{\mathbf{A}}_i^{w_i} \Rightarrow \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_n^{w_n} \hat{\mathbf{A}}^w \mathbf{A}_1^{w_1} \dots \mathbf{A}_{i-1}^{w_{i-1}}$$

for all $1 \leq i \leq n$.

To see why this will suffice, suppose first that

$$\mathbf{S}^\$ \Rightarrow \mathbf{A}_1^{w_1} \dots \mathbf{A}^{w_{i-1}} \mathbf{A}_i^{w_i} \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_n^{w_n} \rightarrow \mathbf{A}_1^{w_1} \dots \mathbf{A}^{w_{i-1}} a \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_n^{w_n}$$

in the original grammar. So $\mathbf{A}_i \rightarrow a$ is in \mathcal{P} . Then in the new grammar

$$\begin{aligned} \mathbf{S}_0^\$ \Rightarrow \tilde{\mathbf{S}}^{w_i} \rightarrow a \hat{\mathbf{A}}_i^{w_i} &\Rightarrow a \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_n^{w_n} \hat{\mathbf{S}}^\$ \mathbf{A}_1^{w_1} \dots \mathbf{A}_{i-1}^{w_{i-1}} \\ &\rightarrow a \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_n^{w_n} \mathbf{A}_1^{w_1} \dots \mathbf{A}_{i-1}^{w_{i-1}}, \end{aligned}$$

hence every cyclic permutation of a word in L is in the new language.

Conversely, suppose $\mathbf{S}_0^\$ \Rightarrow a \mathbf{B}_1^{v_1} \dots \mathbf{B}_n^{v_n}$ and that this subderivation does not start with $\mathbf{S}_0^\$ \rightarrow \mathbf{S}^\$$. Then the subderivation begins with $\mathbf{S}_0^\$ \rightarrow \tilde{\mathbf{S}}^\$ \Rightarrow \tilde{\mathbf{S}}^u \rightarrow a \hat{\mathbf{A}}^u$ for some $u \in \mathcal{I}^*$, $\mathbf{A} \in \mathcal{N}$. Once a ‘hatted’ symbol has been introduced, the only way to get rid of the hat is via the production $\hat{\mathbf{S}}^\$ \rightarrow \varepsilon$. Hence we must have $\hat{\mathbf{A}}^u \Rightarrow \mathbf{B}_1^{v_1} \dots \mathbf{B}_j^{v_j} \hat{\mathbf{S}}^\$ \mathbf{B}_{j+1}^{v_{j+1}} \dots \mathbf{B}_n^{v_n}$ for some $0 \leq j \leq n$ (with the subword before or after $\hat{\mathbf{S}}$ being empty if $j = 0$ or $j = n$ respectively).

But then

$$\begin{aligned} \mathbf{S}^\$ \Rightarrow \mathbf{B}_{j+1}^{v_{j+1}} \dots \mathbf{B}_n^{v_n} \mathbf{A}^u \mathbf{B}_1^{v_1} \dots \mathbf{B}_j^{v_j} \\ \rightarrow \mathbf{B}_{j+1}^{v_{j+1}} \dots \mathbf{B}_n^{v_n} a \mathbf{B}_1^{v_1} \dots \mathbf{B}_j^{v_j} \end{aligned}$$

and so if a word is produced by the new grammar, some permutation of that word is in L .

We finish by giving the inductive proof of the equivalence of (1) and (2). For the base case, we have $\mathbf{A}^w \Rightarrow \mathbf{B}^u \mathbf{C}^v$ if and only if at some point in the parse tree, there is a production $\mathbf{X}^t \rightarrow \mathbf{Y}^t \mathbf{Z}^t$, with $\mathbf{A}^w \Rightarrow \mathbf{X}^t$, $\mathbf{Y}^t \Rightarrow \mathbf{B}^u$ and $\mathbf{Z}^t \Rightarrow \mathbf{C}^v$. The productions in these last three subderivations are all of the form $\mathbf{D} \rightarrow \mathbf{E}^f$ or $\mathbf{D}^f \rightarrow \mathbf{E}$, so they are equivalent to $\hat{\mathbf{X}}^t \Rightarrow \hat{\mathbf{A}}^w$, $\hat{\mathbf{B}}^u \Rightarrow \hat{\mathbf{Y}}^t$ and $\hat{\mathbf{C}}^v \Rightarrow \hat{\mathbf{Z}}^t$. Also $\mathbf{X} \rightarrow \mathbf{Y} \mathbf{Z}$ if and only if $\hat{\mathbf{Y}} \rightarrow \mathbf{Z} \hat{\mathbf{X}}$ and $\hat{\mathbf{Z}} \rightarrow \hat{\mathbf{X}} \mathbf{Y}$. Putting these together, we have $\mathbf{A}^w \Rightarrow \mathbf{B}^u \mathbf{C}^v$ if and only if

$$\hat{\mathbf{B}}^u \Rightarrow \hat{\mathbf{Y}}^t \rightarrow \mathbf{Z}^t \hat{\mathbf{X}}^t \Rightarrow \mathbf{C}^v \hat{\mathbf{A}}^w$$

and

$$\hat{\mathbf{C}}^v \Rightarrow \hat{\mathbf{Z}}^t \rightarrow \hat{\mathbf{X}}^t \mathbf{Y}^t \Rightarrow \hat{\mathbf{A}}^w \mathbf{B}^u,$$

as required.

Now for $k > 2$, suppose our statement is true for $n < k$. Then $\mathbf{A}^w \Rightarrow \mathbf{A}_1^{w_1} \mathbf{A}_2^{w_2} \dots \mathbf{A}_k^{w_k}$ if and only if for each $1 \leq i \leq k$ there are $\mathbf{X}_i, \mathbf{Y}_i, \mathbf{Z}_i \in \mathcal{N}$ and $t \in \mathcal{I}^*$ such that $\mathbf{X}_i \rightarrow \mathbf{Y}_i \mathbf{Z}_i$ and for some $1 \leq j \leq k$ either

$$\mathbf{A}^w \Rightarrow \mathbf{A}_1^{w_1} \dots \mathbf{A}_{i-1}^{w_{i-1}} \mathbf{X}_i^t \mathbf{A}_j^{w_j} \dots \mathbf{A}_k^{w_k},$$

with $\mathbf{Y}_i^t \Rightarrow A_i^{w_i}$ and $\mathbf{Z}_i^t \Rightarrow \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_{j-1}^{w_{j-1}}$, or

$$\mathbf{A}^w \Rightarrow \mathbf{A}_1^{w_1} \dots \mathbf{A}_j^{w_j} \mathbf{X}_i^t \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_k^{w_k},$$

with $\mathbf{Y}_i^t \Rightarrow \mathbf{A}_{j+1}^{w_{j+1}} \dots \mathbf{A}_{i-1}^{w_{i-1}}$ and $\mathbf{Z}_i^t \Rightarrow \mathbf{A}_i^{w_i}$.

We will consider only the second of these, as it is the slightly more complicated one and the first is very similar. The right hand side of the displayed subderivation has fewer than k terms, so by our assumption, this subderivation is valid if and only if

$$\hat{\mathbf{X}}_i^t \Rightarrow \mathbf{A}_{j+1}^{w_{j+1}} \dots \mathbf{A}_k^{w_k} \hat{\mathbf{A}}^w \mathbf{A}_1^{w_1} \dots \mathbf{A}_{i-1}^{w_{i-1}}.$$

But this, together with $\mathbf{Y}_i^t \Rightarrow \mathbf{A}_{j+1}^{w_{j+1}} \dots \mathbf{A}_{i-1}^{w_{i-1}}$ and $\mathbf{Z}_i^t \Rightarrow \mathbf{A}_i^{w_i}$, is equivalent to

$$\hat{\mathbf{A}}_i^{w_i} \Rightarrow \hat{\mathbf{Y}}^t \rightarrow \mathbf{Z}^t \hat{\mathbf{X}}^t \Rightarrow \mathbf{A}_{i+1}^{w_{i+1}} \dots \mathbf{A}_k^{w_k} \hat{\mathbf{A}}^w \mathbf{A}_1^{w_1} \dots \mathbf{A}_{i-1}^{w_{i-1}}.$$

□

Next, we now show that when L is context-free, L_σ is indexed. Since Brandstädt proved that the class of context-free languages is not closed under C^k for all $k \geq 3$, and

$$C^k(L) = \bigcup_{\sigma \in S_k} \sigma(L) = \bigcup_{1 \leq \ell \leq k} \bigcup_{\sigma \in S_\ell} L_\sigma,$$

we have that for all $k \geq 3$ there exist permutations $\sigma \in S_k$ such that L_σ are not context-free for some context-free language L .

Proposition 6. *Let $\tau \in S_\ell$ be a permutation. If L is context-free, then L_τ is indexed.*

Proof. The proof is based partly on a similar idea to the proof of Theorem 5, except that since we are splitting our words up into ℓ subwords rather than only two, we need to consider skeletons with frames having more complicated shapes than just a single edge.

For $w = w_1 \dots w_\ell \in L$ with all w_i non-empty, let x_i be the first symbol of w_i , and consider a parse tree skeleton with frame consisting of the unique paths from the start symbol \mathbf{S}^\S to each x_i for $2 \leq i \leq \ell$ (these paths will generally overlap). An example is shown in Figure 4.

A skeleton B defines a sublanguage of L – namely all those words which can be generated by completing B into a full parse tree – as well as a fixed ℓ -partition of words in this language. Let $L(B)$ be the set of all ‘partitioned words’ $w_1 | \dots | w_\ell$ (where $w_1 \dots w_n \in L$) generated from the skeleton B . Let $L_\tau(B)$ be the set of all words $w_{\tau(1)} \dots w_{\tau(\ell)}$ such that $w_1 | \dots | w_\ell \in L(B)$. Then L_τ is the union of all the (infinitely many) languages $L_\tau(B)$.

Our grammar for L_τ will be based on constructing all possible $(\ell - 1)$ -leaved skeletons in L . There are finitely many possible shapes for the frames of these skeletons, and we will construct one grammar for each tree-shape with $\ell - 1$ leaves.

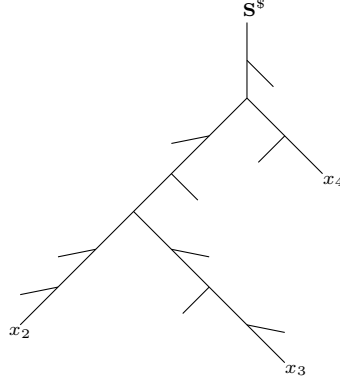


FIGURE 4. Skeleton for L_τ .

We shall abuse standard terminology somewhat, by referring to an edge as a *leaf* if one of its vertices is a leaf, and talking about *parent*, *child* and *sibling* edges, the usual convention being to use these terms for vertices.

Let T be a tree-shape with $\ell - 1$ leaves. We construct a grammar Γ_T that will produce $\bigcup_B L_\tau(B)$, where B ranges over all T -skeletons for words in L . First place an ordering e_1, e_2, \dots, e_N on the set $E(T)$ of edges of T (where $N = 2\ell - 3$), such that e_1 is the edge with one vertex at the root, and for some $1 \leq m \leq N$ the edges e_1, \dots, e_m are all non-leaves, while the remaining edges are leaves. Choose the ordering such that a parent edge always comes before its children, and a left sibling always comes before its right sibling. Also fix an ordering of the degree 3 vertices in T , which we refer to as *branch points* (there are $\ell - 2$ of these).

Let L be generated by a context-free grammar Γ in Chomsky normal form with nonterminals \mathcal{N} , terminals \mathcal{X} and productions \mathcal{P} . Our indexed grammar Γ_T is defined using the following symbols:

- terminals \mathcal{X} ,
- flags $\mathcal{F} = \{\$, \#_i, A_\alpha, A_\omega, A_L, A_R, a_\omega \mid 1 \leq i \leq N, \mathbf{A} \in \mathcal{N}, a \in \mathcal{X}\}$,
- start symbol \mathbf{S}_0 and remaining nonterminals

$$\{\mathbf{A}_e \mid \mathbf{A} \in \mathcal{N}, e \in E(T)\} \cup U \cup \overline{U},$$

where

$$U = \{\mathbf{X}_s, \mathbf{X}_{s,B}, \mathbf{X}_{s,BC}, \mathbf{M}, \mathbf{M}_{ij} \mid s, i \in \{1, \dots, \ell\}, j \in \{1, \dots, k_i\}, \mathbf{B}, \mathbf{C} \in \mathcal{N}\}$$

(k_i will be defined later) and $\overline{U} = \{\overline{\mathbf{C}} \mid \mathbf{C} \in U\}$.

We begin by constructing a potential T -skeleton by replacing each edge e of T by a valid edge-skeleton P_e in Γ . We will check the consistency of the branch points (degree 3 vertices of T) later. For now the initial symbols of all the edge-skeletons other than the one for the initial edge e_1 (which has initial symbol \mathbf{S}_0) will be chosen non-deterministically. We do not need to remember the labels of the vertices on the frames of the edge-skeletons P_e ,

other than the initial and terminal vertices. We can therefore create and store all the important information about our T -skeleton using the following rules. The subscripts α and ω denote the beginning and end respectively of an edge-skeleton P_e , while the subscripts L and R record whether a vertex lies to the left or right of its frame. [Note: Throughout this proof, we will sometimes use brackets around nonterminals to increase legibility. The brackets have no meaning in the grammar.]

- (1) $\mathbf{S}_0 \rightarrow (\mathbf{S}_{e_1})^{S_\alpha}$,
- (2) $\mathbf{A}_e \rightarrow (\mathbf{B}_e)^{C_R} \mid (\mathbf{C}_e)^{B_L}$ for each edge e and production $\mathbf{A} \rightarrow \mathbf{BC}$ in \mathcal{P} ,
- (3) $\mathbf{A}_{e_i} \rightarrow (\mathbf{B}_{e_{i+1}})^{B_\alpha \#_i A_\omega}$ for each $\mathbf{A}, \mathbf{B} \in \mathcal{N}$ and $1 \leq i \leq m$,
- (4) $\mathbf{A}_{e_i} \rightarrow (\mathbf{B}_{e_{i+1}})^{B_\alpha \#_i a_\omega}$ for each production $\mathbf{A} \rightarrow a$ in \mathcal{P} , $\mathbf{B} \in \mathcal{N}$ and $m+1 \leq i \leq N-1$,
- (5) $\mathbf{A}_{e_N} \rightarrow (\mathbf{M})^{\#_N A_\omega}$ for each $\mathbf{A} \in \mathcal{N}$.

After a sequence of these productions, terminating with (5), the string produced is a single nonterminal \mathbf{M} with flag

$$\#_N \omega_N v_N \alpha_N \#_{N-1} \dots \#_2 \omega_2 v_2 \alpha_2 \#_1 \omega_1 v_1 \alpha_1 \$.$$

The section of the flag in between $\#_i$ and $\#_{i-1}$ contains the information about the path-skeleton P_{e_i} . The symbols α_i and ω_i correspond to the initial and final vertices of P_{e_i} respectively, and we have $\alpha_i = B_\alpha$ for some nonterminal \mathbf{B} (in particular, $\alpha_1 = S_\alpha$), while $\omega_i = A_\omega$ for some nonterminal \mathbf{A} if $1 \leq i \leq m$, and $\omega_i = a_\omega$ for some terminal a otherwise (that is, if e_i is a leaf). The v_i are words in $\{A_L, A_R \mid A \in \mathcal{N}\}^*$ encoding – in reverse – the sequence of vertices off the main path in P_{e_i} , with the subscripts L and R denoting a vertex lying to the left or right of the path respectively.

Having produced a flag corresponding to a potential T -skeleton, we now need to check that this is indeed a valid T -skeleton in Γ . The only potential problems are at the branch points. If e_p is an edge with left child e_q and right child e_r , then we need to check that \mathcal{P} contains a production $\mathbf{A} \rightarrow \mathbf{BC}$, where $\omega_p = A_\omega$, $\alpha_q = B_\alpha$ and $\alpha_r = C_\alpha$. In order to do this, we create a ‘check symbol’ \mathbf{X}_i for each branch point.

- (6) $\mathbf{M} \rightarrow \overline{\mathbf{M}} \mathbf{X}_1 \dots \mathbf{X}_{\ell-2}$.

Let e_p be the edge coming down into the i -th branch point and let e_q and e_r be its left and right children respectively. Recall that we have chosen our ordering on the edges in such a way that $p < q < r$. Call a nonterminal \mathbf{A} *f-ready*, for some flag f , if $\mathbf{A}^g \rightarrow \mathbf{A}$ for all $g \in \mathcal{F} \setminus \{f\}$. Now \mathbf{X}_i checks for a valid production at the i -th branch point. Informally, the idea is that \mathbf{X}_i searches the flag for the symbols α_r, α_q and ω_q (which will occur in that order) and stores C and B , where $\alpha_r = C_\alpha$, $\alpha_q = B_\alpha$, finally outputting the empty word if and only if $\mathbf{A} \rightarrow \mathbf{BC}$ is in \mathcal{P} , where $\omega_p = A_p$. Formally, this is achieved via the following productions, requiring quite a few extra nonterminals:

- (7) \mathbf{X}_i is $\#_r$ -ready, with $(\mathbf{X}_i)_{\#_r} \rightarrow \overline{\mathbf{X}}_i$,

- (8) $\overline{\mathbf{X}}_i$ is ready for the next α -subscripted flag (which will be α_r), and $(\overline{\mathbf{X}}_i)^{C_\alpha} \rightarrow \mathbf{X}_{i,C}$,
- (9) $\mathbf{X}_{i,C}$ is $\#_q$ -ready, with $(\mathbf{X}_{i,C})^{\#_q} \rightarrow \overline{\mathbf{X}}_{i,C}$,
- (10) $\overline{\mathbf{X}}_{i,C}$ is ready for the next α -subscripted flag (which will be α_q), and $(\overline{\mathbf{X}}_{i,C})^{B_\alpha} \rightarrow \mathbf{X}_{i,BC}$,
- (11) $\mathbf{X}_{i,BC}$ is $\#_p$ -ready, with $(\mathbf{X}_{i,BC})^{\#_p} \rightarrow \overline{\mathbf{X}}_{i,BC}$,
- (12) The next flag will be ω_p , so $\overline{\mathbf{X}}_{i,BC}$ can now finally check the validity of the branch point, by having productions $(\overline{\mathbf{X}}_{i,BC})^{A_\omega} \rightarrow \varepsilon$ for all productions $\mathbf{A} \rightarrow \mathbf{BC}$ in \mathcal{P} .

These productions ensure that once the symbols \mathbf{X}_i are introduced, the derivation will never terminate unless the potential T -skeleton in the flag has valid connections at all branch points, and hence is a valid T -skeleton, in which case all the symbols \mathbf{X}_i produce the empty word.

Finally, we ‘unpack’ the T -skeleton from the flag, to produce words $\tau(w)$, where w is a word in L arising from the T -skeleton. Let e_i^L and e_i^R be the left and right hand sides of the edge e_i respectively. Consider the ‘outline’ of T , which is a directed path $o(T)$ drawn around the outside of T , beginning on the left hand side of the root and ending on the right hand side of the root, divided into labelled segments corresponding to the edge-sides.

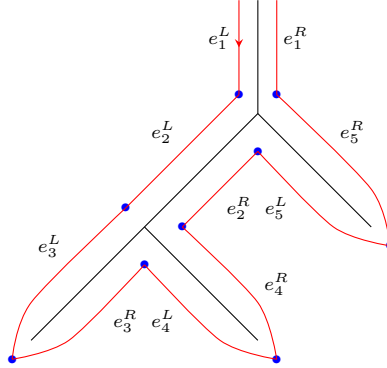


FIGURE 5. Outline.

The segment of $o(T)$ lying between the $(i-1)$ -th and i -th leaves of T corresponds to w_i (here we regard the root as both the 0-th and ℓ -th leaf). For $1 \leq i \leq \ell$, let k_i be the length of this segment. We write $w_i \sim f_1 \dots f_{k_i}$ if the labels on the segment of $o(T)$ corresponding to w_i , in order, are f_1, \dots, f_{k_i} . For $1 \leq i \leq \ell$, define $\rho_i : \{1, \dots, k_i\} \rightarrow \{1, \dots, N\}$ and $d_{ij} \in \{L, R\}$ such that if $w_i \sim f_1 \dots f_{k_i}$, then $f_j = e_{\rho_i(j)}^{d_{ij}}$. Then if w is a word in L produced from our T -skeleton, we have $w = w_1 \dots w_\ell$, where w_i arises from the edge-sides $e_{\rho_i(1)}^{d_{i1}}, \dots, e_{\rho_i(k_i)}^{d_{ik_i}}$ in that order. Note that $d_{i1} = R$ for all $i \geq 2$. We prepare to unpack the flag in the correct order to produce $\tau(w)$ as follows:

- (13) $\overline{\mathbf{M}} \rightarrow \overline{\mathbf{M}}_{\tau(1)1} \dots \overline{\mathbf{M}}_{\tau(1)k_1} \dots \overline{\mathbf{M}}_{\tau(\ell)1} \dots \overline{\mathbf{M}}_{\tau(\ell)k_\ell}$.

And finally, the actual unpacking occurs, using the following productions.

- (14) $\overline{\mathbf{M}}_{ij}$ is $\#_{\rho_i(j)}$ -ready, with $(\overline{\mathbf{M}}_{ij})^{\#_{\rho_i(j)}} \rightarrow \mathbf{M}_{ij}$,
- (15) $(\mathbf{M}_{ij})^{a_\omega} \rightarrow a\mathbf{M}_{ij}$ if $d_{ij} = R$ and $(\mathbf{M}_{ij})^{a_\omega} \rightarrow \varepsilon$ if $d_{ij} = L$,
- (16) $(\mathbf{M}_{ij})^{A_L} \rightarrow \mathbf{M}_{ij}\tilde{\mathbf{A}}$ if $d_{ij} = L$, and $(\mathbf{M}_{ij})^{A_L} \rightarrow \mathbf{M}_{ij}$ if $d_{ij} = R$,
- (17) $(\mathbf{M}_{ij})^{A_R} \rightarrow \tilde{\mathbf{A}}\mathbf{M}_{ij}$ if $d_{ij} = R$, and $(\mathbf{M}_{ij})^{A_R} \rightarrow \mathbf{M}_{ij}$ if $d_{ij} = L$,
- (18) $(\mathbf{M}_{ij})^{B_\alpha} \rightarrow \varepsilon$ and $(\mathbf{M}_{ij})^{A_\omega} \rightarrow \varepsilon$,
- (19) for all $\mathbf{A} \in \mathcal{N}$, $\tilde{\mathbf{A}}$ is $\$$ -ready and $\tilde{\mathbf{A}}^\$ \rightarrow \mathbf{A}$,
- (20) all productions in \mathcal{P} .

The productions allow $\overline{\mathbf{M}}_{ij}$ to find the section of the flag corresponding to the edge $e_{\rho_i(j)}$, from which the j -th segment of w_i arises, and then unpack the relevant parts (i.e. the vertices from the left or right side) of that section in the correct direction. The flag contains the information about each edge-skeleton P_e in reverse order. For subwords generated by e^L , this is the ‘wrong’ order and so we need to unpack the left-hand vertices of P_e to the right, while for subwords generated by e^R , this is the correct order and so we unpack the right-hand vertices of P_e to the left. A flag a_ω belongs to the right side of its edge, by our convention for the partition defined by the tree-shape T . When we reach a symbol B_α , we have finished recovering the relevant segment, and so we output ε . Finally, we produce an appropriate subword of a word in L using productions from \mathcal{P} . Once we have produced a word consisting entirely of terminals, we have $w_{\tau(1)} \dots w_{\tau(\ell)}$ for some partitioned word $w_1 | \dots | w_\ell$ in $L(B)$, where B is the T -skeleton encoded in the flag. All such words can be produced in this way, and so the language generated by Γ_T is indeed the union of all $L_\tau(B)$ with B a T -skeleton in Γ . Hence L_τ is the union of finitely many indexed languages and is thus itself indexed. \square

Our main result follows immediately.

Corollary 7. *Let $\sigma \in S_k$ be any permutation. If L is context-free, then $\sigma(L)$ is indexed.*

Corollary 8. *Let k be a positive integer. If L is context-free, then $C^k(L)$ is indexed (and context-free if $k = 1, 2$).*

REFERENCES

- [1] Alfred V. Aho. Indexed grammars—an extension of context-free grammars. *J. Assoc. Comput. Mach.*, 15:647–671, 1968.
- [2] Andreas Brandstädt. Closure properties of certain families of formal languages with respect to a generalization of cyclic closure. *RAIRO Inform. Théor.*, 15(3):233–252, 1981.
- [3] Werner Damm. The IO- and OI-hierarchies. *Theoret. Comput. Sci.*, 20(2):95–207, 1982.
- [4] Werner Damm and Andreas Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Inform. and Control*, 71(1-2):1–32, 1986.
- [5] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.

- [6] A. N. Maslov. The cyclic shift of languages. *Problemy Peredači Informacii*, 9(4):81–87, 1973.
- [7] Takeshi Oshiba. Closure property of the family of context-free languages under the cyclic shift operation. *Electron. Commun. Japan*, 55(4):119–122, 1972.

SCHOOL OF MATHEMATICS AND STATISTICS, UNIVERSITY OF ST ANDREWS, NORTH HAUGH, ST ANDREWS KY16 9SS, SCOTLAND
E-mail address: `t.brough@st-andrews.ac.uk`

MATHEMATICS DEPARTMENT, UNIVERSITY OF NEUCHÂTEL, RUE EMILE - ARGAND 11, CH-2000 NEUCHÂTEL, SWITZERLAND
E-mail address: `laura.ciobanu@unine.ch`

SCHOOL OF MATHEMATICAL AND PHYSICAL SCIENCES, THE UNIVERSITY OF NEWCASTLE, CALLAGHAN NSW 2308, AUSTRALIA
E-mail address: `murray.elder@newcastle.edu.au`